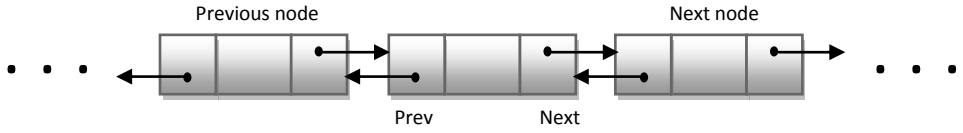
القوائم Lists

2.5.3 القائمة المرتبطة الثنائية Doubly Linked List:

المفاهيم الأساسية والأسماء التعريفية Basic concepts and nomenclature :

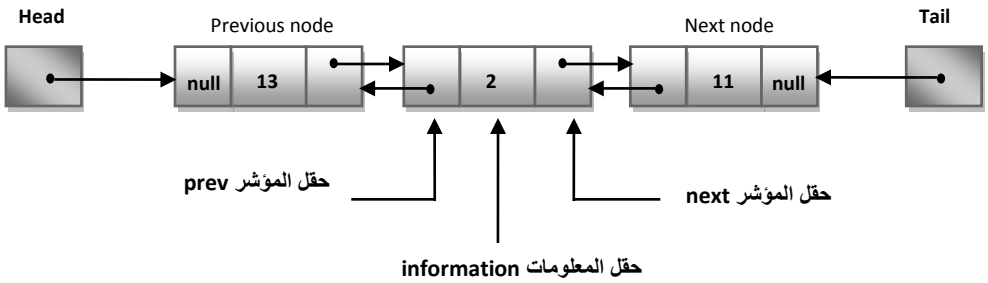
كل سجل record أو عقدة node يتضمن ثلاث حقول three fields :

- **الحقل الأول the first field** : يتضمن معلومات عن العنصر item ويعرف **بالبيانات data** أو **المعلومات information** أو **القيمة value**.
- **الحقل الثاني the second field** : يتضمن العنوان address للعقدة التالية next node في القائمة .
- **الحقل الثالث the third field** : يتضمن العنوان address للعقدة السابقة previous node في القائمة .



المؤشر الرأسي Head pointer : يستخدم ليحمل العنوان address لأول عنصر في القائمة list . أيضاً، العنصر الأول للقائمة المرتبطة يحمل قيمة null في حقل prev pointer ليدل على بداية القائمة list .

المؤشر النهاية Tail pointer : يستخدم ليحمل العنوان address لأخر عنصر في القائمة list . أيضاً، العنصر الأخير للقائمة المرتبطة يحمل قيمة null في حقل next pointer ليدل على نهاية القائمة list .



إعلان القائمة المرتبطة الثنائية Declaration of a Doubly Linked List:

نفرض أننا نريد تخزين قائمة list من الأعداد الصحيحة int's ، إذاً القائمة المرتبطة يمكن أن تكون معلنة كالتالي:

```
typedef struct nodetype
{
 int info;
 struct nodetype *next;
 struct nodetype *prev;
} node;
node *head;
node *tail;
```

تكوين قائمة فاضية Creating an empty list:

- في الإعلان السابق، المتغيران الرأس head والنهاية tail كل منهما معلن كمؤشر للعقدة node.
- المتغيران الرأس head و النهاية tail لم يتم تهيئتهما حتى الآن.
- المتغير الرأس head يستخدم ليؤشر على العنصر الأول في القائمة list والمتغير النهاية tail يستخدم ليؤشر على العنصر الأخير في القائمة list.
- لكون القائمة ستكون فاضيه empty في البداية، المتغيران الرأس head و النهاية tail لابد من تخصيص قيمة مبدئية لهما للإشارة على أن القائمة فاضيه empty.

```
void createemptylist(node *head, node *tail)
{
  head=NULL;
  tail=NULL;
}
```

الإدخال في القائمة المرتبطة الثنائية Insertion in a Doubly Linked List :

لإدخال عنصر item في القائمة:

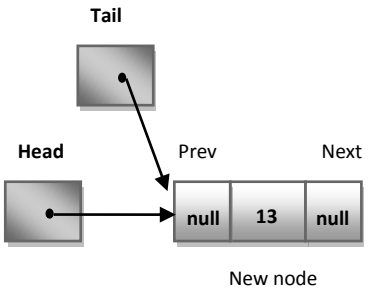
- يتم أولاً تكوين عقدة جديدة new node ،
- يحدد العنصر ليتم إدخاله في حقل المعلومات info field للعقدة الجديدة new node ،
- بعد ذلك يضع العقدة الجديدة new node في الموقع الملائم position بتعديل المؤشرات pointers .

الإدخال في القائمة يمكن أن يحدث في المواقع التالية:

1. <u>الإدخال في بداية القائمة Insertion at the Beginning of the List :</u>

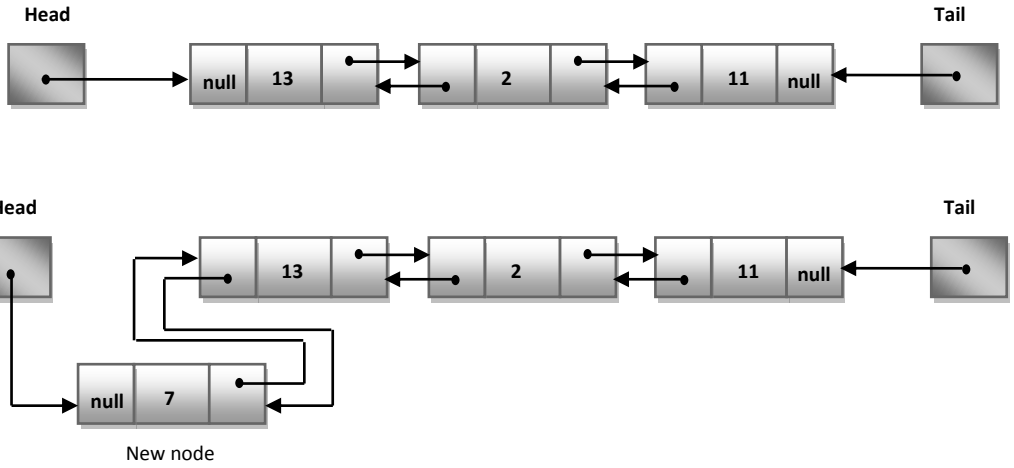
أولاً، نختبر إذا كانت القائمة المرتبطة linked list فاضية empty أم لا . إذا كانت فاضية empty، يتم إدخال العنصر item وهو بذلك يعتبر أول عنصر تم إدخاله في القائمة، وبذلك يكون هناك عنصر واحد فقط داخل القائمة. ويتم ذلك بإجراء الخطوات التالية:

- يتم تخصيص الـ null لحقل المؤشر next و حقل المؤشر prev للعقدة الجديدة new node.
- يتم تخصيص العنوان address للعقدة الجديدة new node لهؤشر الرأس head و مؤشر النهاية trail.



إذا لم تكن القائمة فاضية empty، يتم إدخال العنصر item قبل أول عنصر في القائمة. ويتم إنجاز ذلك عن طريق الخطوات التالية:

- يتم تخصيص NULL لحقل المؤشر prev للعقدة الجديدة new node.
- يتم تخصيص قيمة الرأس head لحقل المؤشر next للعقدة الجديدة new node.
- يتم تخصيص العنوان address للعقدة الجديدة new node لحقل المؤشر prev للعقدة التي يؤشر عليها حالياً بمؤشر الرأس head.
- يتم تخصيص العنوان address للعقدة الجديدة new node للمؤشر الرأس head.



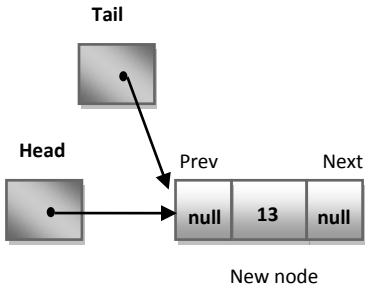
برمجياً، كلا الحالتين متكافئتين . وذلك لأن في الحالتين الخطوة الأول يتم فيها تخصيص قيمة الرأس head (null أو غير ذلك) لحقل المؤشر next للعقدة الجديدة new node.

```
void insertatbeginning(node *head, int item)
{
 node *newNode;

/* allocate memory for the new node and initialize the data in it*/
 newNode= malloc(sizeof(node));
 newNode->info=item;

 /* assign the value of head to the "next" of newNode*/
 newNode->next=head;
 head->prev= newNode

/* assign the address of newNode to first node Previously ,head and
   tail */
 head->prev=newNode;
 head=newNode;
 if(newNode->next=NULL)
  tail=newNode;

/* assign NULL value to the "prev" of newNode*/
 newNode->prev=NULL;

}
```

2. الإدخال في نهاية القائمة Inserting at the End of the List :

أولاً، نختبر إذا كانت القائمة المرتبطة linked list فاضية empty أم لا . إذا كانت فاضية empty، يتم إدخال العنصر item وهو بذلك يعتبر أول عنصر تم إدخاله إلى القائمة، وبذلك يكون هناك عنصر واحد فقط داخل القائمة . ويتم إنجاز ذلك عن طريق الخطوات التالية:

- يتم تخصيص الـ null لحقل المؤشر next و حقل المؤشر prev للعقدة الجديدة new node.
- يتم تخصيص العنوان address للعقدة الجديدة new node لمؤشر الرأس head و مؤشر النهاية trail.



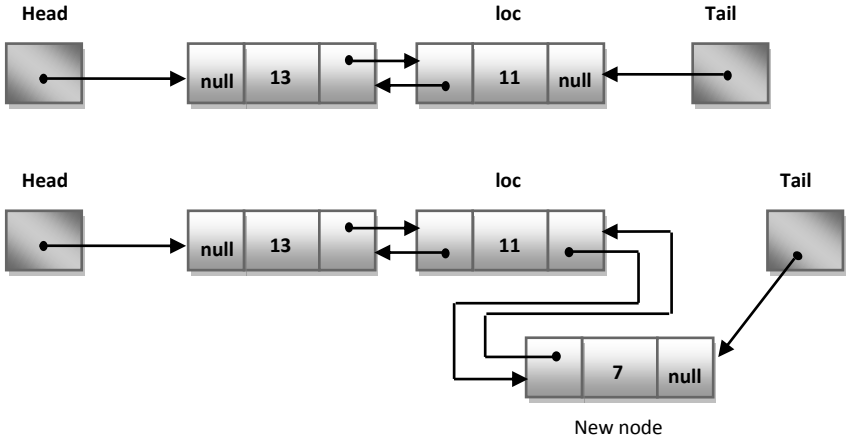
إذا كانت القائمة ليست فاضية not empty ، يتم الوصول إلى العنصر الأخير في القائمة last item ، ثم يتم إدخال العقدة الجديدة new node كأخر عنصر last item في القائمة، ويتم ذلك بإجراء الخطوات التالية:

- يتم تخصيص الـ null لحقل المؤشر next للعقدة الجديدة new node.
- يتم تخصيص قيمة tail لحقل المؤشر prev للعقدة الجديدة new node.
- يتم تخصيص العنوان address للعقدة الجديدة new node لحقل المؤشر next للعقدة الأخيرة last node الحالية وذلك بواسطة المؤشر tail.
- أخيراً، يتم تخصيص العنوان address للعقدة الجديدة new node للمؤشر tail.

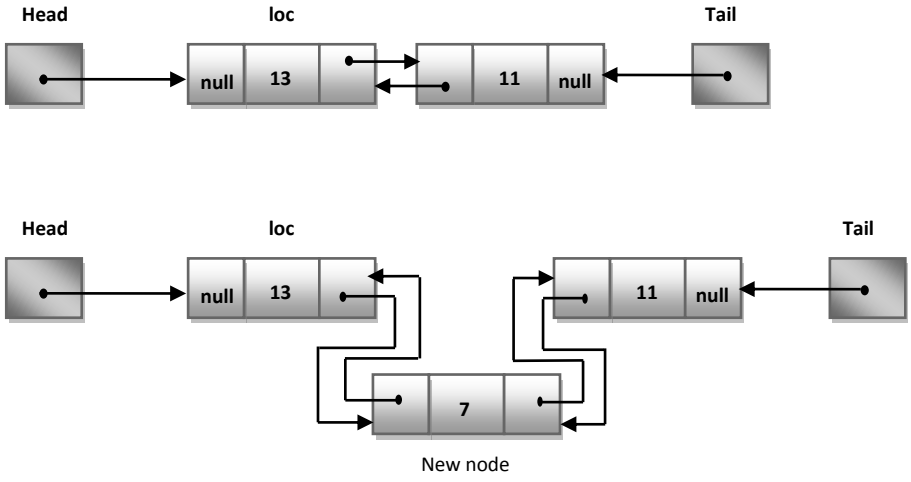
```
void insertatend (node *head, node *tail, int item)
{   node *newNode;
   newNode = malloc(sizeof(node));
   newNode ->info = item;
    if (head == NULL)
    {
       newNode ->next = newNode ->prev=NULL;
       head = tail = newNode;
    }else
    {
       newNode ->next=NULL;
       newNode ->prev=tail;
       tail->next= newNode;
       tail= newNode;
    }
}
```

3. الإدخال بعد عنصر معطى Inserting after Given Element :

لإدخال عنصر جديد بعد عنصر معطى given element. أولاً، يتم تحديد الموقع للعنصر المعطى في القائمة وليكن loc .



(a)



(b)

```
void insertafterelement (node *head, node *tail, int item, node *loc)
{
    node *newNode;
    newNode=malloc(sizeof(node));
    newNode->info = item;
    if(loc->next == NULL)
    {
        newNode->next = NULL;
        loc->next = newNode;
        newNode->prev = tail;
        tail = newNode;
    }
    else
    {
        newNode->prev = loc;
        newNode->next = loc->next;
        (loc->next)->prev = newNode;
        loc->next = newNode;
    }
}
```

لإلغاء عنصر من القائمة المرتبطة الثنائية Deleting an Element from a Doubly Linked List :

لإلغاء عنصر من القائمة، أولاً يتم وضع المؤشرات pointers بشكل صحيح بحيث يتم إلغاء العقدة node من القائمة، العقدة الملغاة لا تزال تشغل موقع في الذاكرة memory ولتحرير هذا الموقع نستخدم دالة (free) . ويتم حذف العقدة node من القائمة من المواقع التالية:

1. الإلغاء من بداية القائمة Deleting from the Beginning of the List :

```
void deletefrombeginning( node *head, node *tail)
{
 node *temp;
 if(head==NULL)
  printf("the list is empty");
 else
 {
   temp=head;
   if(head==tail)       /*one element only*/
    head=tail=NULL;
   else
    {
     (temp->next)->prev=NULL;
      head=temp->next;
    }
   free(temp);
 }
}
```

2. الإلغاء من نهاية القائمة Deleting from the End of the List :

للإلغاء من نهاية القائمة، لابد من الوصول إلى العنصر ما قبل الأخير في القائمة . بعد ذلك يمكن إلغاء العنصر الأخير last item .

```
void deletefromend( node *head, node *tail)
{
 node *temp;
 if(head==NULL)
  printf("the list is empty");
 else
 {
   temp=tail;
   if(head==tail)    /*one element only*/
    head=tail=NULL;
   else
   {
    temp->prev->next=NULL;
    tail=temp->prev;
   }
   free(temp);
 }
}
```

3. الإلغاء بعد عنصر معطى Deleting after a Given Element :

لإلغاء العنصر بعد عنصر معطى prev ، يتم إجراء الخطوات التالية:

```
void deleteafterelement (node *head, node *tail, node *loc)
{
 node *temp;
 if (loc->next==NULL)   /*element 'after' not found*/
   printf("element 'after' not found");
```

```
 else
 {
   temp = loc->next;
   loc->next = temp->next;
   if(temp->next == NULL)
     tail = loc;
   else
     (temp->next)->prev = loc;
   free(temp);
 }
}
```

4. إلغاء القائمة بالكامل Deleting the Entire List :

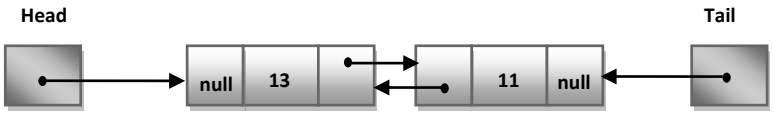
قبل إنهاء البرنامج، لابد من إلغاء القائمة بالكامل وذلك لأن الذاكرة محجوزة من قبل القائمة list ولابد من إعادة الذاكرة المحجوزة لنظام التشغيل os .

```
void deletelist(node *head, node *tail)
{
 node *temp;
 while(head!=NULL)
 {
   temp=head;
   head=(head)->next;
   free(temp);
 }
 tail=NULL;
}
```

البحث في قائمة مرتبطة أحادية Searching a Doubly Linked List :

نعبر القائمة من البداية، ونقارن كل عنصر في القائمة بالعنصر المعطى بواسطة المتغير temp والمطلوب البحث عنه.

item=11




```
node *search(node *head, int item)
{
 node *temp;
 temp=head;
 while(head!=NULL) &&(temp->info!=item))
   temp=temp->next;
 return temp;
}
```